

LiLo: A Live Looping Musical Interface

and

Inlayers: A Composition using LiLo for the Princeton Laptop Orchestra, So Percussion, Matmos, and Electric Guitar

Michael W. Hammond

Music Department
Princeton University
Princeton, NJ, USA

May 2009

A Senior thesis submitted to the Music Department, Princeton University in partial fulfillment of the requirements for the degree of Bachelor of Arts in Music.

I hereby declare that I am the sole author of this thesis. I authorize Princeton University to lend this to other institutions or individuals for the purpose of scholarly research.

I further authorize Princeton University to reproduce this thesis by photocopying or by other means in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Author's Note

This document serves as a supplement to the main body of the thesis, which can be found at the following website:

<http://www.music.princeton.edu/~hammond/LiLo/>

Abstract

LiLo is a live looping playground created as a performance and compositional aid for musicians. In many ways, LiLo resembles a traditional looping device. With the touch of a button, loops can be created, removed, re-recorded, and overdubbed. At this minimal level, LiLo acts as an open-source replacement for your Boss RC-20. However, LiLo expands on these basic features in a number of important ways. First, LiLo allows the performer to chop up their loops into a number of smaller “chunks,” which can be rearranged, reversed, randomized, or rotated left and right. Each chunk has its own volume, pitch, and rate that can be edited in LiLo’s various edit modes. Furthermore, the loops can be “chunked” according to several methods:

- Manual chunking
- Automatic chunking into equal-sized divisions
- Feature-based chunking (chunking according to degrees of variance among features derived from either time-domain signal data or FFT spectral data, including centroid, RMS, spectral flux, and spectral roll-off)

The performer may also add multiple iterations (called “voices”) of a buffer. Each voice can be chunked independently, providing endless possibilities for creating new textures. Additionally, LiLo is completely customizable, meaning that presets can be written to call up a specific voice/chunk arrangement at any point during performance. This combination of features gives the performer unprecedented control over his or her audio loops.

At first blush, looping appears to be an inherently stagnant form of composition. A loop is created and then repeats itself, ad infinitum. Unfortunately,

this is a fairly apt description of the history of looping in music. When Steve Reich and his contemporaries began using tape loops in the 1960s, there was little alternative: Real-time manipulation of a tape loop would often result in the destruction of the tape. But even with the advent of digital audio and signal processing, the vast majority of commercial looping devices continue to behave as if they were miniature reel-to-reels. Partly because of this, composers who loop (some out of necessity) often feel the need to obfuscate the “looping” qualities of the loop. Skilled loopers such as Bill Frisell and Andrew Bird have become experts at hiding their loops behind flowing sonic textures. While this is a completely valid and effective form of composition in its own right, LiLo hopes to reinvent the looper as a dynamic and interactive tool, a playground for live or private performance.

While this is an admittedly ambitious goal, LiLo’s combination of intuitive design and flexible control points the way toward a not-so-distant future where looping is itself a form of expression on par with traditional instruments. Already, LiLo has been successfully utilized in a variety of performance settings. The piece composed for this thesis project, *Inlayers*, enacts a sort of looping jam session by giving LiLo to each of the 30 members of the Princeton Laptop Orchestra. In the piece, players load pre-recorded loops and manipulate them according to cued instructions, though they are free to improvise within certain constraints. Another successful format has been the “looping duo,” which consists of an instrumentalist and laptopist. The former loops his or her sound in LiLo with a MIDI stomp box or other interface while the latter “performs” the loops.

Contents

1. About.....	7
1.1 Development.....	7
1.2 References.....	7
2. Using LiLo: A Beginner's Guide.....	8
2.1 Running LiLo.....	8
2.2 Make some noise.....	8
2.3 Other Methods of Chunking.....	10
2.4 Editing Rates, Rate Edit Mode, and Pitch Edit Mode.....	11
2.5 Presets.....	12
2.6 Munge Mode.....	12
3. Composing for LiLo and PLOrk.....	14
3.1 <i>Inlayers</i> : Lilo's first performance.....	14
3.2 The Text Message System.....	15
4. Performances and Recordings.....	18
4.1 The Kitchen, NYC.....	18
4.1 Richardson Auditorium, Princeton, NJ.....	18

1. About

1.1 Development

LiLo was developed primarily in ChuckK, an audio programming language developed by Ge Wang at Princeton. The visual interface was written in Processing, a language that harnesses Java's graphics engine. For more information, visit these sites:

<http://www.chuck.cs.princeton.edu>

<http://www.processing.org>

To communicate between ChuckK and Processing, LiLo makes use of the Open Sound Control (OSC) Protocol.

1.2 References

In the LiLo source code (which can be found at the website mentioned in the Author's Note), there are comments that reference sources for portions of the code. These are too numerous to mention here. However, I should note that my advisor Daniel Trueman was a constant source of information and guidance throughout this project and that without his help, LiLo would be but a fraction of its current self.

2. Using LiLo: A Beginner's Guide

2.1 Running LiLo

First things first. Download the files from the following website:

<http://www.princeton.edu/~hammond/thesis/>

Before you can run LiLo, be sure you've downloaded Chuck:

<http://chuck.cs.princeton.edu/>

Now, in the downloaded directory, you will see a Bash script titled "LiLo." To run LiLo, double click the script file. The LiLo GUI should pop up shortly. First, run your mouse over the LOAD_FILE menu. You should see a list of files. If not, relaunch.

2.2 Make some noise

To load a sound clip, click on a file in the LOAD_FILE menu. When the file begins to load, you will see the Chunk Edit Window:

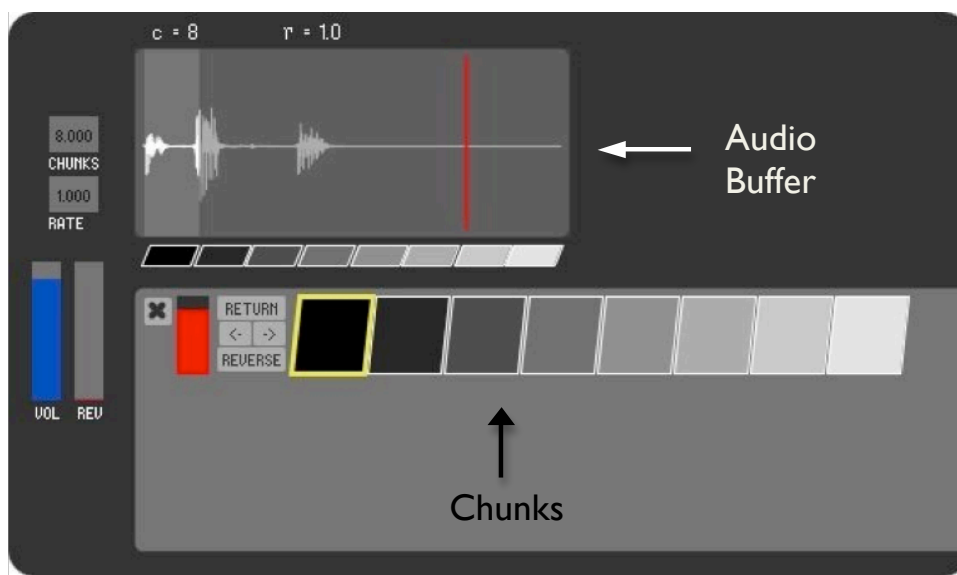


Figure 1. The Audio Buffer and Chunk Window

The Chunk Edit Window illustrates how the sound file is divided into smaller rhythmic chunks. The number of chunks can be edited using the “CHUNKS” text field just to the left of the buffer. The file is loaded into ChuckK, but you won’t hear anything until you manually bring up the gain for each chunk. This can be done by entering Gain Edit Mode (using the TAB key – the backdrop will turn purple) and clicking on each chunk to reveal translucent volume sliders. To easily bring up all chunk gains, hold the COMMAND key while editing the chunk gains.

To navigate through the chunks, use the arrow keys or the mouse. You can delete chunks (“x” or DELETE key), change a chunk to another one (number keys), reverse the chunks (“r”), scramble them (“s”), change their rates (SHIFT keys, or using the “RATE” text field to the right of the audio buffer), and so on. In addition to the keyboard shortcuts, each voice has a *Voice Control Panel*, that lets you rotate, reverse, and descramble the chunks. Also in the Voice Control Panel is a handy slider to change the volume of the voice and a close box to quickly remove the voice.



Figure 2. The Voice Control Panel

An easy way to add a new chunk is to grab it from the smaller set of chunks displayed just below the buffer and move it to the voice you want. Additionally, the GUI is designed so that you can intuitively drag-and-drop chunks from one voice to another, or below the last existing voice to create a new voice immediately. A handy list of all the important keyboard shortcuts can be found at the top of the GUI. In

addition, you can add a new voice (essentially another layer of the same audio buffer) with the “n” key. (If you don’t like the fact that all your new voices are silent and every time you re-chunk a voice, it becomes silent, go to the upper-right hand corner of the GUI and edit the “Incoming_Chunk_Vol” slider).

2.3 Other Methods of Chunking

Equal-length chunks are great and all, but maybe you’d rather chunk by some other means. Easy! To manually chunk your loop, hold down the OPTION key and click in the audio buffer window to add a new chunk division.

One of the more exciting ways to chunk loops is to use one of the four features displayed in the *Feature Chunking Control Panel*:

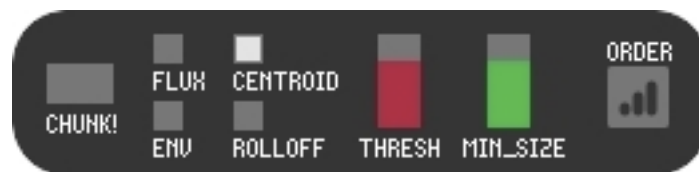


Figure 3. Feature Chunking Control Panel

(NOTE: If you’re running the *LiLo_for_PLOrk* script, you won’t see the Feature Chunking Panel and its place will have a window to receive text messages from the server.) To chunk a voice, simply select the desired feature. Centroid and rolloff are generally best at distinguishing between frequencies while flux and envelope follower track amplitude shifts and will chunk according to significant changes in the power of a signal. As you adjust the threshold slider and min_chunk_size slider, your voice will be chunked automatically. If you’d like to chunk an existing voice with the same parameter settings, simply select the voice

and hit the “Chunk!” button. The “Order” button automatically puts the chunks in order according to the feature values associated with each chunk.

2.4 Editing Rates, Rate Edit Mode, and Pitch Edit Mode

To change the rate of all the chunks in a voice, use the left and right SHIFT keys. These allow you to quickly change the rate of a voice by a factor of 2. To change the rate of a voice to something other than a factor of 2, use the rate text field to the left of the audio buffer. By default, all rates change instantaneously, but if you’d rather sweep gradually to the new rate, edit the “Rate_Ramp_Time” text field in the upper-right hand corner. This sets the time (in seconds) for the voice to reach its new rate destination.

To edit the pitches of individual chunks, TAB over to Pitch Edit Mode (green backdrop). Here you can grab the individual sliders and yank them around to change the pitch of each chunk (within an octave). (NOTE: You won’t have access to Pitch Edit Mode unless you’ve clicked the LiLo_5Voices script. This is because Pitch Edit Mode requires a special build of ChuckK that incorporates a multichannel version of the LiSa unit generator object. Running this version of ChuckK will allow you to pitch shift individual voices, but keep in mind you’ll be limited to five measly voices!)

Now, if you’re in Pitch Edit Mode, TAB over once more to reveal Rate Edit Mode (brown backdrop). In Rate Edit Mode, you can edit individual chunk rates, with a catch: every time you edit a chunk rate, all other chunk rates are altered so as to maintain the original length of the voice. This means that all changes in Rate Edit

Mode are non-time-destructive. So you can go into a rate-changing frenzy and your loops will still line up! (At least, after a sync or two.)

2.5 Presets

Just below the LOAD_FILE menu is a PRESETS menu. Choosing a new preset will automatically add multiple voices to a loop, changing the rates and chunk orderings of the voices to match the specifications of that preset. The built-in presets have been hand selected by the composer for use with the built-in audio files to be played on *Inlayers*, but you can create your own presets by altering the source code.

2.6 Munge Mode

Munge Mode allows you to munge the output of *all* the currently running loops.

Entering Munge Mode (CAPS LOCK) will bring up the lovably off-kilter Munge

Window:

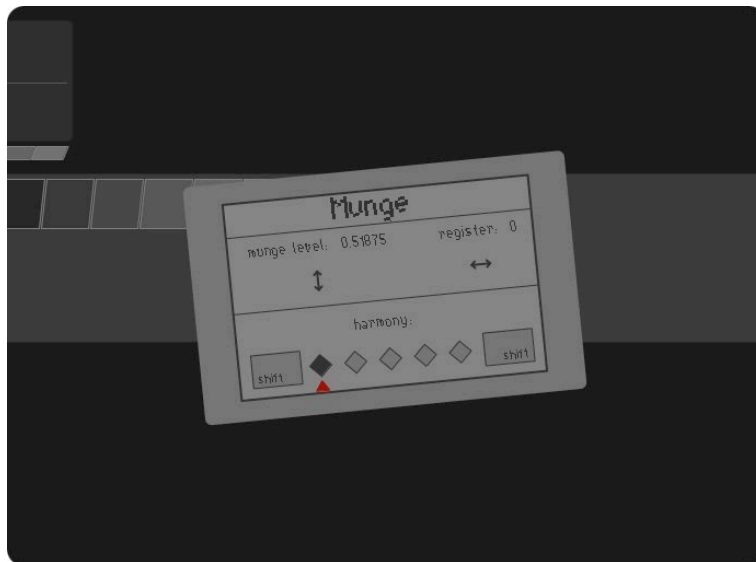


Figure 4. The Munge Window

Munging is controlled by the built-in accelerometer of your Apple computer. The x-axis controls register while the y-axis controls the mix of the munged sound relative to the un-munged LiLo output. In turn, these changes are reflected in the orientation of the Munge Window. The vertical orientation of the Munge Window follows your movements right and left while the size of the grey backdrop is a virtual representation of your up and down movements. The harmonies of the munging are hard-coded in LiLo (8ve, m3rd, M3rd, P4, and P5, respectively). To select among the pre-determined harmonies, use the SHIFT keys.

3. Composing for LiLo and PLOrk

3.1 *Inlayers*: Lilo's first performance

Although LiLo was originally conceived as a performance and compositional tool for individual performers, it has proven to be quite powerful in ensemble settings as well. The first piece written for LiLo – *Inlayers*, composed by the author of the software – takes advantage of this by spreading LiLo out among the 30 members of the Princeton Laptop Orchestra. The ensemble is divided into three sections, with each section assigned a different set of loops to open in LiLo. The players receive further instructions in a text message window that appears when a special version of LiLo is launched (this can be accessed by double clicking the *LiLo_for_PLOrk* script in the download folder). Performance duties of the players include: rate-shifting, editing chunk gains, editing chunk pitches, editing the number of chunks, and munging. For the performance, special munge harmonies were loaded into LiLo and the server sent messages over a local network causing the munge window to re-orient itself according to which register was desired by the composer. The following instructions were given to the performers and may help explain their duties in this section:

During the munging section of the piece, you have 3 goals: to keep the Munge Window centered on the screen, to keep the rounded rectangular outline inside

the smaller gray boundary, and to follow the red triangle which designates the munge harmony (to change your munge harmony use the left and right SHIFT keys). The munge window will try to run away but don't let it! (NOTE: if server is not running, the munge window won't actually try to run away, but it may still be helpful to practice wobbling your laptop around to get used to the sensitivity of the response).

Although sections of *Inlayers* consist solely of PLOrk playing LiLo, the piece is a collaborative effort with the groups Matmos and So Percussion, who are working with PLOrk in the spring of 2009. Using samples of acoustic and electric guitars recorded by the composer, Matmos constructed a variety of rhythms that feature prominently in the piece. The question of how to rhythmically synchronize two electronic groups quickly became a crucial issue. Although collectively synchronizing the entire PLOrk ensemble over a network was auditioned early on, this was dropped in favor of a more organic, and sometimes more unpredictable, method wherein each individual player is responsible for synching himself or herself to Matmos's rhythms.

3.2 The Text Message System

The idea for implementing a messaging system in LiLo came from a piece by Dan Trueman, the *Network Cyclor*, originally performed at the inaugural PLOrk concert. Sending messages over a network (in English) can be a simple and effective way to communicate with the members of a large laptop ensemble. In this case, it

was an extremely valuable aspect of the compositional process as well. Each week, as PLOrk would rehearse my piece, I would try out new text cues. Initially, I used a texting conductor interface (which can be opened from the download folder by clicking *serverText*). The interface allows the conductor to have the ability to access message presets or to send custom messages. In this case, I would go in with a new set of cues each week with a certain idea of how things might sound. When things inevitably did not sound how I had imagined, I would improvise some custom messages, eventually landing on a set of techniques that work for PLOrk's large, spatialized sound.

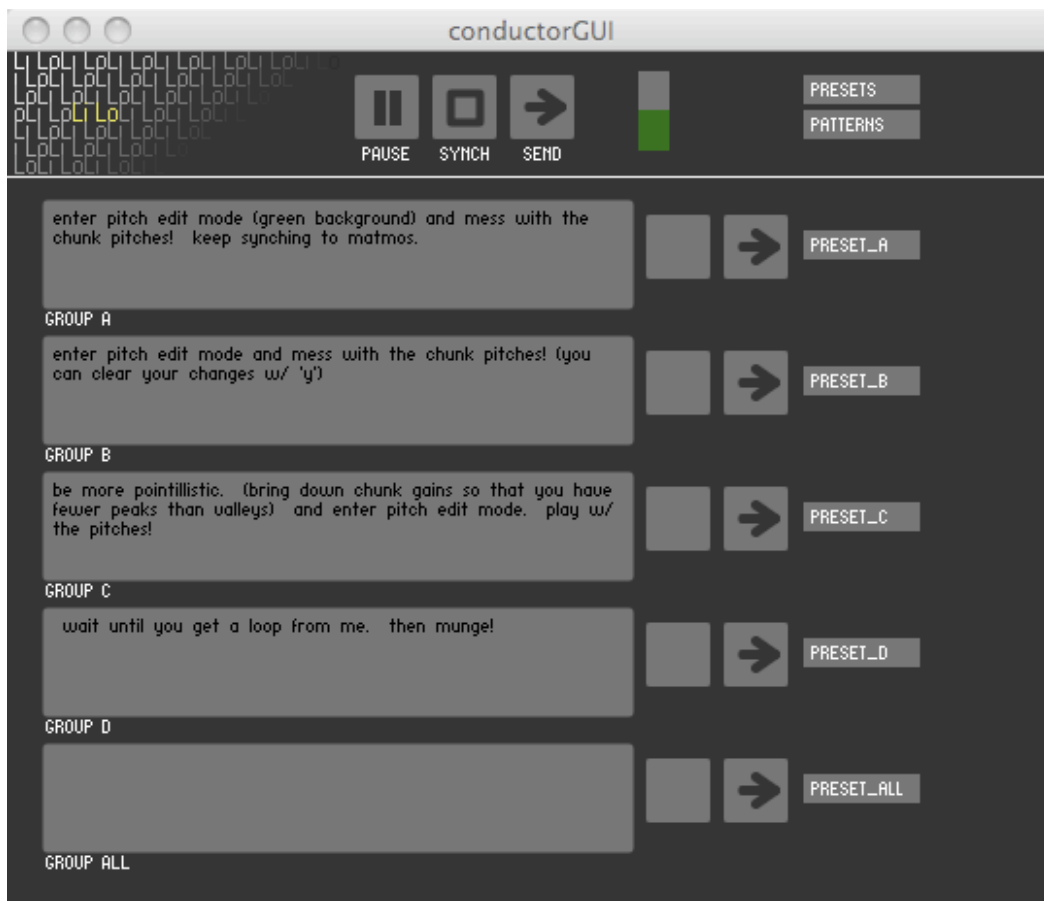


Figure 5. The Conductor GUI

Once the text cues were solidified, the graphic text server interface was ditched in favor of a more practical approach: sending out pre-determined text cues after a stomp box MIDI event. Doing this allowed me to perform my electric guitar parts while continuing to send out texts. The MIDI stomp box was already programmed to loop the guitar sound, so it was an obvious choice to have the text cues available at the stomp of a button.

4. Performances and Recordings

4.1 The Kitchen, NYC

[April 10 and 11, 2009] *Inlayers* was performed by PLOrk, Matmos, So Percussion, and the author on electric guitar at The Kitchen in Manhattan. Enclosed are two DVDs, one of Friday evening's performance and one of Saturday evening. PLOrk member Raymond Weitekamp is live looping the guitar sound.

4.1 Richardson Auditorium, Princeton, NJ

[May 16, 2009] upcoming